(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 901 096 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
19.04.2006 Bulletin 2006/16

(51) Int Cl.:
G06K 15/00 (2006.01) *H04N 1/58* (2006.01)

(21) Application number: **98307091.3**

(22) Date of filing: **03.09.1998**

(54) Peeker detection and correction

Erkennung und Korrektion von Farblücken

Détection et correction d'espacement de couleurs

(84) Designated Contracting States:
DE GB

(30) Priority: **03.09.1997 US 922781**

(43) Date of publication of application:
10.03.1999 Bulletin 1999/10

(73) Proprietor: **Adobe Systems, Inc.**
San Jose,
California 95110 (US)

(72) Inventors:
- **Dermer, Richard A.**
  Issaquan,
  Washington 98029 (US)
- **Becker, Douglas R.**
  Ayer,
  Massachusetts 01432 (US)
- **Mercer, Dennis**
  Auburn,
  New Hampshire 03032 (US)

(74) Representative: **Rocke, Carsten et al**
Müller-Boré & Partner
Grafinger Strasse 2
81671 München (DE)

(56) References cited:
EP-A- 0 618 718
WO-A-95/20796

Description

Background of the Invention

[0001] The invention relates to modifying a color page described in a page description language so that the page can be correctly trapped.

[0002] In the printing industry, press registration is the accurate positioning of two or more colors of ink on a printed sheet. When the colors on a sheet are in register, all colors appear precisely where intended, without gaps between colors or overlap of colors. Misregistration can be caused by a number of factors, including errors in platemaking and film handling prior to platemaking, poorly-maintained printing presses or poorly-trained operators, paper inaccuracy or instability, and lack of environmental controls. One result of misregistration is unprinted paper showing through between colors, where no unprinted area was intended. Even a small registration error can result in a thin white line showing between colors on a sheet.

[0003] Trapping is the process of compensating for press misregistration by intentionally overlapping colors prior to printing. For example, a spread expands the color of a foreground image to overlap with the background color. Conversely, a choke enlarges the background color to overlap with the foreground image. The area of color added to create overlap, called a trap, provides a degree of protection against gaps created by misregistration. Although the trap color may be the same as the background or foreground color, it need not be. Trapping involves deciding where to place traps, and setting the size and color of traps so as to correct for possible misregistration while being minimally noticeable to the human eye.

[0004] It is conventional in the electronic publishing industry to design pages using a page description language (PDL), such as the Adobe ® PostScript ® language, to describe a page using resolution-independent elements. In PostScript, a path is a collection of ordered directed connected straight line segments, referred to as edges. A path, which appears as a shape on the page, must be closed. Each edge in a path has a from point and a to point which define the position and direction of the edge. The to point of an edge in a path is the same point as the from point of the subsequent edge in the path, and the to point of the last edge in the path is the same point as the from point of the first edge in the path. Edges in a path may intersect.

[0005] An object may consist of one or more paths, all filled or stroked in a single color. An object has a unique sequence number, also referred to as a z-order or paint order, that is shared by each of the paths in the object. The sequence number of an object specifies when the object will be drawn on the page in relation to other objects. The object with the lowest sequence number will be drawn first, followed by the object with the next lowest sequence number. This sequential drawing of objects may result in one object being partially or entirely occluded by other objects with higher sequence numbers.

[0006] To render a PostScript page (by which is meant a PostScript language description of a page) on an output device such as a printer, the page must first be rasterized, that is, the vector representation of the page must be converted into a two-dimensional array of pixels. The resolution of an output device is defined by the number of pixels on the output medium of the output device per unit distance or per unit area.

[0007] A PostScript page description may be analyzed to produce a new PostScript page description which contains traps. The new PostScript page description may then be used to render the page, with traps, according to conventional methods. Alternatively, the PostScript page may first be rasterized, after which traps are generated in rasterized form by analyzing the rasterized page.

[0008] WO 95/20796 discloses a process for applying traps to a page by using a trapping analyser that generates trap vectors in accordance with a set of desired trapping rules which take into account existing traps in the proximity of other page objects. The trapping process described in this document includes raster based and vector based solutions. In a raster based solution, proposed trap pixels are evaluated for conflicts with existing objects and other traps to determine if a given pixel is to be included in a current trap.

[0009] EP 0 618 718 A1 discloses a method and apparatus for compensation for misregistration of printing plates in printing of polychromatic document pages or images, in which a trapping map image is superimposed upon a structured graphic image representing the layout of a document page or image from which it is derived, so as to prevent light leaks and other errors at the boundaries between colour regions within the image.

[0010] It is the object of the invention to provide a method for preparing a resolution independent representation of a region for trapping providing a basis for a more accurate positioning of traps.

[0011] This object is solved according to the invention by a method having the features disclosed in claim 1. Preferred embodiments are disclosed in the dependent claims.

Summary of the Invention

[0012] A peeker is a gap between two edges in vector representation that is narrower than a device pixel at the resolution of the output device on which the vectors are to be rendered. The invention modifies a vector image to be rendered in rasterized form, so that traps may be correctly generated wherever peekers exist. In particular, the invention corrects vector space color transitions so that they match color transitions in rasterized form.

[0013] In one aspect, the invention prepares a resolution-independent representation of a region for trapping. Specifically, a peeker is found between a first edge in the

region and a second edge in the region, and an abutting color that will abut a side of the first edge between the first edge and the second edge when the first edge is rendered in rasterized form is identified. The color of the first edge on the side of the first edge between the first edge and the second edge is then changed to the abutting color. The region is then trapped and rendered on an output device, such as a printer.

[0014] In another aspect, the invention identifies an exterior color of a resolution-independent edge in a path based on information derived from the interior colors of other edges in other paths. The exterior color so identified may then be advantageously used in peeker detection and color correction, and in trapping.

[0015] Among the technical advantages of the invention are one or more of the following.

[0016] One advantage of the invention is that it performs peeker color correction on objects while they are in vector form, prior to rasterization. This provides a basis for more accurate positioning of traps than trapping based on the rasterized page by itself. Also, because the invention provides a basis for resolution-independent trapping, pages which are trapped according to the invention may be rendered at a number of resolutions without the need for re-trapping.

[0017] A further advantage of the invention is that use of the vector representation preserves information about objects that is lost during rasterization, allowing more complete and varied processing of the edge table.

[0018] Furthermore, vector space is a more efficient representation of the rendered appearance of an area than raster space; a color transition many hundred pixels in length can be described in one small data structure in vector form. This storage efficiency is critical to obtaining high throughput and efficient trapping performance.

[0019] Other features and advantages of the invention will become apparent from the following description and from the claims.

Brief Description of the Drawings

[0020]

FIG. 1 is a flow diagram of a method of the invention.
FIGs. 2a-2b are diagrams of PostScript paths.
FIG. 2c is a diagram of a PostScript path partially occluding another PostScript path.
FIG. 2d is a diagram of the visible and invisible edges resulting from the application of a hidden line removal algorithm to two PostScript paths within one tile.
FIG. 2e is a diagram of a PostScript path within a tile, and of a vector used to identify the exterior color of the PostScript path.
FIGs. 3a-3c are flow diagrams of methods used to identify the exterior colors and exterior sequence numbers of edges within a tile.
FIG. 4a is a diagram of two PostScript paths between which a peeker exists.
FIG. 4b is a diagram of a rasterization of parts of the two objects in FIG. 4a at the location of the peeker.
FIG. 5 is a flow diagram of Phase One of peeker detection and correction.
FIG. 6a is a diagram of a first peeker rectangle drawn around a first edge according to the method of FIG. 4.
FIG. 6b additionally shows a second edge originating in the first peeker rectangle, and a resulting split point generated on the first edge.
FIGs. 7a-f are flow diagrams of Phase Two of peeker detection and correction.
FIG. 8a is a diagram of two peeker squares drawn around the endpoints of an edge longer than twice the peeker distance.
FIG. 8b additionally shows two additional edges, each of which has an endpoint in each peeker square.
FIG. 8c is a diagram of a peeker rectangle drawn around an edge whose length is less than or equal to twice the peeker distance.
FIG. 8d additionally shows an additional edge having both endpoints within the peeker rectangle.
FIG. 9 illustrates a computer and computer elements suitable for implementing the invention.

Detailed Description

[0021] Referring to FIG. 1, a method for trapping a PDL page detecting and correcting peeker colors of the invention first obtains a resolution-independent description of a page (e.g., in PostScript format) (step 2), and then transforms the page description into a list of objects (step 4). The method then identifies and removes hidden edges from the page (step 6), and identifies the exterior color and exterior sequence number of each edge on the page (step 8). The method next identifies peekers using the vector representation of the objects on the page and information about the resolution at which the page will be rendered in rasterized form on an output device (step 10).

[0022] Then, the method changes the exterior color of each edge at which a peeker exists to the color which will abut the exterior side of the edge when rendered on the output device (step 12). This changes the color transitions between edges in vector space so that they match the color transitions between edges in rasterized form.

[0023] Finally, trapping is performed on the page, using the corrected color information, using any of a variety of trapping methods (step 14).

[0024] The method of FIG. 1 is now described in more detail.

[0025] Referring to FIG. 2a, a PostScript path 16 has ordered edges E1, E2, E3, E4, E5, E6, and E7. The first edge E1 has a from point P1 and a to point P2. The remaining edges, similarly, have from and to points which define the shape of the path 16 in a counter-clockwise direction, ending at point P1. As each edge of path 16 is traced from its from point to its to point, the interior of path 16 is defined to be on the left side of the edge; the

exterior of path 16 is defined to be on the right side of the edge. Each edge in path 16 has an interior color that specifies the color on the interior (left) side of the edge. The interior color of each edge of path 16 is yellow, causing the interior of path 16 to appear yellow when rendered on an output device.

[0026] Referring to FIG. 2b, a PostScript path 18 has ordered edges E8, E9, E10, E11, and E12. The first edge E8 has a from point P3 and a to point P4. The remaining edges, similarly, have from and to points which trace the path 18 in a counter-clockwise direction, ending at point P3. The interior color of each edge is magenta, causing the interior of path 18 to appear magenta when rendered on an output device.

[0027] Objects on a PostScript page may overlap. Therefore, some object edges may be obscured by other objects and may therefore not be visible when the page is rendered on an output device. If path 18 has a higher sequence number than path 16, then the paths will appear as shown in FIG. 2c when rendered by an output device. As shown in FIG. 2c, part of the interior region of path 16 is occluded by part of the interior region of path 18. Edges E3, E4, E5, and E6 of path 16 are entirely occluded by path 18, and parts of edges E2 and E7 of path 16 are occluded by path 18.

[0028] A PostScript page is transmitted to a PostScript-capable device, such as a PostScript printer or imagesetter. A trapping engine transforms the PostScript page into a set of objects appearing on the page, using conventional methods. Each object consists of one or more paths sharing a common unique sequence number.

[0029] Each edge in a path has a number of attributes. Each edge has a from point and a to point, which define the position and direction of the edge. The original interior color and effective interior color of an edge are initialized to the value of the interior color of the edge. The owning or interior sequence number of an edge is the sequence number of the path to which the edge belongs. The original exterior color, effective exterior color, and exterior sequence number of an edge are described below. A child edge (described below) also contains a reference to the child edge's parent edge.

[0030] After the trapping engine transforms the PostScript page into a set of objects, the page is divided into small rectangular tiles. Although the invention will be described as operating on such tiles, it may also operate on a page which has not been divided into tiles. Each tile boundary has a from point, a to point, and an interior color.

[0031] The edges in a tile which will be visible when the page is rendered are recorded in an edge table. Referring to FIG. 3a, the trapping engine creates an empty edge table for a tile and adds the tile boundaries of the tile to the edge table (step 20). Next, the object o with the lowest sequence number in the tile is selected (step 21), and the original exterior colors of object o's edges are initialized to a value of unknown. The edges of object o are then added to the edge table (step 23).

[0032] A hidden line removal method is then applied to the edge table. The hidden line removal method splits each edge in the edge table into two smaller edges at each point at which the edge intersects an edge of another object. Each edge resulting from a split inherits the original interior color and interior sequence number of the edge from which it was derived.

[0033] For example, referring to FIG. 2d, edge E2 of object 16 is split into edge E13 (with endpoints P2 and P7) and edge E14 (with endpoints P7 and PS), because edge E2 crosses a boundary of object 18 at point P7. Similarly, edge E7 of object 16 is split into edge E15 (with endpoints P6 and P8) and edge E16 (with endpoints P8 and P1) because edge E7 crosses the boundary of object 18 at point P8.

[0034] A complex point is a point at which more than two visible edges intersect. Referring to FIG. 2d, for example, point P7 is a complex point. A spline is a contiguous series of connected edges which do not cross an object boundary. A spline either begins and ends at a complex point or begins and ends at a single point. The hidden line removal method divides each object into splines, and identifies each of the edges of each object as being either visible or invisible. The edges of each invisible spline are then removed from the edge table.

[0035] For example, as a result of applying the hidden line removal method to the tile containing object 16 and object 18, object 16 is divided into two splines: one spline (S1) consisting of edges E13, E1, and E16, and one spline (S2) consisting of edges E14, E3, E4, E5, E6, and E15. Similarly, object 18 is divided into two splines: one spline (S3) consisting of edges E21 and E17, and one spline (S4) consisting of edges E18, E19, and E20. Splines S1, S3, and S4 are visible, as shown in FIG. 2c. Spline S2 is not visible.

[0036] After the hidden line removal method has identified the visible and invisible edges in a tile, the first visible edge e in object o is selected (step 26). If the original exterior color of edge e is unknown (step 30), then it is determined whether either endpoint of edge e is at a complex point (step 32). If either endpoint of edge e is at a complex point, then the exterior color and exterior sequence number of edge e are assigned values, according to the method of FIG. 3b (step 50). Steps 26-50 are then then repeated for each remaining visible edge in the tile.

[0037] Referring to FIG. 3b, the original exterior color and exterior sequence number of an edge e that intersects with a complex point p are assigned values as follows. During hidden edge removal, a list of all visible edges connected to point p is maintained. First, the visible edges connected to point p are sorted (step 52). Edges are sorted in order of increasing angular direction (in a clockwise direction) with respect to point p. For purposes of sorting, the direction of each edge is treated as if the edge points away from point p. For example, the edges at point P8 (FIG. 2d) are sorted as follows: E20, E21, E16. Note that edge E15 is not included in the list because it was previously identified as invisible by the hidden line removal method. Next, if e points into point p, then ee is

selected as the edge previous to e in the sorted list, with wraparound (step 53). If e points away from point p, then ee is selected as the edge following e in the sorted list, with wraparound (step 54). In other words, the edge previous to the first edge in the list is the last edge in the list, and the edge after the last edge in the list is the first edge in the list. For example, the edge previous to E21 is E20, and the edge, in the sort order, after E16 is E20.

[0038] Next, the original exterior color of e is set to the original color of ee on the side of ee facing the exterior of e, and the exterior sequence number of e is set to the sequence number of ee on the side of ee facing the exterior of e (step 55). For example, referring to to FIG. 2d, if p is point P8 and e is edge E21, then ee is edge E7. The interior of edge E7 faces the exterior of edge E21. Therefore, the original exterior color of edge E21 is set to the original interior color of edge E7 (yellow), and the interior sequence number of edge E7 is copied into the exterior sequence number of edge E21.

[0039] The original exterior color and exterior sequence number of edge e are then propagated to the other edges in edge e's spline (step 57). The method then proceeds to step 34 in FIG. 3a (step 60).

[0040] After the method of FIG. 3a has completed operation, the method of FIG. 3c is applied to all edges whose exterior color is still unknown. Referring to FIG. 2e, object 16 is the sole object within tile 19, with tile boundaries TBE1, TBE2, TBE3, and TBE4. Because none of the edges of objec.t 16 intersects with the edges of any other objects or with any tile boundaries, the exterior color of all of the edges of object 16 will remain "unknown" after applying the method of FIG. 3a.

[0041] Referring to FIG. 3c, for each edge e with an unknown original exterior color (step 100), a vector v is constructed perpendicular to e, emanating from the midpoint of e, pointing towards the exterior of e (step 105). For example, referring to FIG. 2e, vector v is constructed perpendicular to edge E1 of path 16. The method next follows v until it intersects either an edge or a tile boundary (referred to as the intersected edge ie) (step 110) that is not in the same object as e (step 115). For example, referring to FIG. 2e, vector v intersects tile boundary TBE2. If v intersects the exterior of ie, then the original exterior color of e is assigned the value of the original exterior color of ie, and the exterior sequence number of e is assigned the value of the exterior sequence number of ie (step 125). Otherwise, the original exterior color of e is assigned the value of the original interior color of ie, and the exterior sequence number of e is assigned the value of the interior sequence number of ie (step 120). The original exterior color and exterior sequence number of e is then propagated to the other edges in e's spline (step 130). For example, referring to FIG. 2e, vector v intersects the interior of TBE2. The original exterior color of edge E1 is therefore assigned the value of the original interior color and the original interior sequence number of tile boundary TBE2, both of which are then propagated to edges E2, E3, E4, E5, E6, and E7.

[0042] Referring to FIG. 4a, in their resolution-independent vector representations, object 200 and object 300 approach each other, but do not touch or overlap, near the top edge 240 of object 200. The interior color of object 200 is cyan, the interior color of object 300 is magenta, and the background color is green. Although not shown, the curve of object 300 is represented in vector form by a number of straight connected edges.

[0043] FIG. 4b shows a possible rasterization of the portions of objects 200 and 300 within the box 310, when rasterized on a device with a resolution smaller than the distance between the top edge 240 of object 200 and the bottom of object 300.

[0044] As shown in FIG. 4b, the pixels representing object 300 touch the pixels representing object 200, even though in their vector representations the objects do not touch or overlap. Therefore, although in the vector representation in FIG. 4a the color on the exterior of edge 240 is green (the background color), in rasterized form the color part of the exterior of edge 240 is magenta. Trapping edge 240 based solely on an exterior color of green will therefore lead to incorrect results. The methods and apparatus of the present invention provide proper trapping in this case, i.e., in the case where the raster representations of two objects touch even though their vector representations do not touch.

[0045] A peeker is a gap between two edges in vector representation that is narrower than a device pixel at the resolution of the output device on which the vectors are to be rendered. If a peeker exists between two edges, the color inside the gap between the two edges will not be visible when the page is rendered in rasterized form. If a peeker exists, the color between the two edges needs to be set to an appropriate color so that correct traps can be generated.

[0046] The peeker distance PD is defined as the maximum distance between two edges for which a peeker will be considered to exist. Trap width is the distance that a trap projects into the darker of two objects between which a trap is required. For a trap width of .25 points, PD is advantageously chosen to be twice the length of the shortest axis (highest dpi) of a pixel on the output device on which the page is to be rendered. PD may also be fixed as (max(h, v) / 1200) * 1, where h is the horizontal resolution of the output device in dots per inch, v is the vertical resolution of the output device in dots per inch, and 1 is the length of the shortest axis of a pixel on the output device. Other methods for calculating PD may also be used. For example, PD may be equal to 1 for resolutions of 0 through 900dpi, 1.5 * 1 for resolutions of 900 through 1800dpi, and (max(h, v) / 1200) * 1 for resolutions greater than 1800dpi. For narrower trap widths, PD should be appropriately scaled down.

[0047] Referring to FIG. 5, Phase One 600 of peeker detection and correction processes each of the edges in a tile as follows. First, an edge e is chosen from the tile's edge table (step 605). Edges may be chosen in any order. If edge e is less than two peeker distances long, then

edge e does not need to be processed by Phase One (step 610). Otherwise, a rectangle r is constructed around edge e as shown in FIG. 6a (FIG. 5, step 620). Edge e has a from point 800 and a to point 805. Rectangle r has corners 820, 825, 835, and 840. Rectangle r is constructed so that PD is the distance between two points in each of the following pairs of points: (810, 820), (810, 825), (815, 835), (815, 840), (810, 800), and (815, 805).

[0048] . Next, all edge endpoints falling within rectangle r, excluding any endpoints created in step 680, are added to a point list (step 630). Next, for each endpoint ep in the point list, a split point 870 on e is identified by drawing a line 860 from ep to e, in a direction perpendicular to e (step 660), as shown in FIG. 6b.

[0049] After split points for e have been identified, selected split points are eliminated so that no two split points are separated by a distance less than PD (step 670). Then, edge e is split into child edges at each split point (step 680). A reference to edge e is stored in each of edge e's child edges. Edge e's interior and exterior color are stored in each of edge e's child edges as the original interior color and original exterior color, respectively, of the child edge (step 690). After being split into child edges, edge e is removed from the edge table and is referred to as a parent edge. For example, referring to FIG. 6b, edge e is split into two child edges. One child edge has a from point at point 800 and a to point at point 870. The other child edge has a from point at point 870 and a to point at point 805.

[0050] If edge e was not the last edge in the tile, (step 700), the method of Phase One 600 is repeated for the remaining edges in the tile. After Phase One 600 has been completed, the method proceeds to Phase Two (step 1000).

[0051] Phase Two processes each of the edges in the edge table of a tile as follows. Referring to FIG. 7a, first a visible edge referred to as a test edge is selected from the edge table (step 1010). Visible edges may be selected as test edges in any order. If the length of the test edge is greater than twice the peeker distance (step 1015), then an edge object pointer tel is set to point to the test edge (step 1025). Referring to FIG. 8a, tel has a from point 1330 and a to point 1340. A square referred to as a from peeker square 1310, with width 2 * PD, is drawn with its center on the from point 1330 of edge tel (step 1030). Two parallel edges of the from peeker square 1310 are parallel to edge tel; the other two parallel edges of the from peeker square 1310 are perpendicular to edge tel.

[0052] Next, a from point list is created by examining the endpoints of each of the edges in the edge table (not including the endpoints of tel), and filling the from point list with each such endpoint which is within the from peeker square 1310 (step 1040). Then, all of the edges to which the points in the from point list belong are stored in a from edge list (step 1050). The steps 1030-1050 are then repeated for the to point 1340 of edge te1, using a to peeker square 1320 (steps 1060-1080).

[0053] After the from edge list and to edge list have been created, a list of candidate peeker edges (the candidate list) is created (steps 1090-1092). First, edges which are members of both the from edge list and the to edge list are added to the candidate list (step 1090). Next, parent edges with a child edge in the from edge list and another child edge in the to edge list are added to the candidate list (step 1092). For example, referring to FIG. 8b, edge 1360 has a from point 1362 in the to peeker square 1320 of edge tel and a to point 1364 in the from peeker square 1310 of edge tel. Edge 1360 is therefore added to edge tel's candidate list at step 1150. Edge 1366 is a parent edge. One child of edge 1366 has a from point 1368 in the from peeker square of edge tel and a to point 1370. The other child of edge 1366 has a from point 1370, and a to point 1372 in the to peeker square of edge tel. The parent edge 1366 is therefore added to edge tel's candidate list at step 1160. Next, the candidates are processed according to the method of FIG. 7c (step 1094).

[0054] Referring to FIG. 7b, if the length of a test edge is less than or equal to 2 * PD (step 1020), then an edge object pointer te2 is set to point to the test edge (step 1100). A single peeker rectangle 1350 is drawn around te2, as shown in FIG. 8c (step 1102). The width of the peeker rectangle 1350 is equal to the length of te2, and te2 divides the peeker rectangle 1350 into equal rectangular halves, each with height PD. A single point list of all endpoints in the peeker rectangle 1350, including the endpoints of te2, is created (step 1104), and a single edge list containing all edges with one or both endpoints in the point list is created (step 1106). The candidate list is then filled with all of the edges in the edge list (step 1108). For example, referring to FIG. 8d, edge 1380 has a from point 1382 and a to point 1384 in the peeker square 1350 of edge te2. Edge 1380 is therefore added to edge te2's candidate list at step 1108. Next, the candidates are processed according to the method of FIG. 7c (step 1094).

[0055] When an edge is placed in the candidate list, the following information about the edge is stored in a data structure: (1) a pointer to the candidate edge in the edge table; (2) the position of the candidate edge's from point in relation to the test edge (left, right, or colinear); (3) the position of the candidate edge's to point in relation to the test edge (left, right, or colinear); and (4) the difference in direction between the candidate edge and the test edge, as measured in radians.

[0056] Referring to FIG. 7c, after the peeker candidate list has been created, each candidate edge in the candidate list is processed. First, candidate edges with at least one endpoint on the left side of the test edge and neither endpoint on the right side of the test edge are processed (step 1120). Specifically, referring to FIG. 7d, the effective interior color of the test edge is set to the interior color of the candidate edge with the highest owning sequence number, from among the candidate edges being processed (step 1140).

[0057] Next, candidate edges with at least one endpoint on the right side of the test edge and neither endpoint on the left side of the test edge are processed (step 1125; FIG. 7e). First, candidate edges from the same object as the test edge are excluded from consideration if they do not point in a direction nearly opposite to the test edge; specifically, they are excluded if the difference in direction is less then $3\pi/4$ radians or greater than $5\pi/4$ radians (step 1150). From the remaining candidate edges, the candidate edge with the highest interior sequence number is identified (step 1155). If the directional difference between the test edge and the identified candidate edge is greater than or equal to $\pi/2$ radians, then the effective exterior color of the test edge is set to the original interior color of the identified candidate edge (step 1160). If the directional difference between the test edge and the identified candidate edge is less than $\pi/2$ radians, the effective exterior color of the test edge is set to the original exterior color of the identified candidate edge (step 1170).

[0058] Next, referring to FIG. 7f, candidate edges which are coterminal with the test edge at both endpoints are processed (step 1130). First, variables MLSN (representing the highest candidate left sequence number encountered so far) and MRSN (representing the highest candidate right sequence number encountered so far) as initialized to -1 (step 1200). Variable TLSN is a pointer to the test edge's left sequence number, and TRSN is a pointer to the test edge's right sequence number (step 1200).

[0059] A candidate c is chosen (step 1202). Variable CLSN is assigned c's left sequence number, and variable CRSN is assigned c's right sequence number (step 1204). If c points in the same direction as the test edge (step 1206), CLSN > TLSN, and CLSN > MLSN, then the test edge's effective left color is assigned c's original left color, and MLSN is assigned the value of CLSN (steps 1220-1224). If c points in the same direction as the test edge (step 1206), CRSN > TRSN, and CRSN > MRSN, then the test edge's effective right color is assigned c's original right color, and MRSN is assigned the value of CRSN (steps 1226-1230).

[0060] If c points in the opposite direction as the test edge (step 1206), CLSN > TRSN, and CLSN > MLSN, then the test edge's effective right color is assigned c's original left color, and MLSN is assigned the value of CLSN (steps 1208-1212). If c points in the opposite direction as the test edge (step 1206), CRSN > TLSN, and CRSN > MRSN, then the test edge's effective left color is assigned c's original right color, and MLSN is assigned the value of CLSN (steps 1214-1218).

[0061] After peeker detection and correction has been performed, trapping can be performed using the effective interior and effective exterior colors of the edges on the page. Trapping will include deciding whether and where to place traps, and, if a trap is to be set, deciding what the color of the trap should be. Typically, a trap engine will examine the effective colors on both sides of each edge in an edge table to determine whether a trap is needed. If a trap is needed, the trap engine will then decide what color the trap should be. Trapping may be performed when the edges are in vector form, or it may be performed on the rasterized page.

[0062] Referring to FIG. 9, the invention may be implemented in digital electronic circuitry or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention may be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a computer processor; and method steps of the invention may be performed by a computer processor executing a program to perform functions of the invention by operating on input data and generating output.

[0063] Suitable processors 1480 include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory (ROM) 1520 and/or a random access memory (RAM) 1510 through a CPU bus 1500. A computer can generally also receive programs and data from a storage medium such as an internal disk 1430 operating through a mass storage interface 1440 or a removable disk 1410 operating through an I/O interface 1420. The flow of data over an I/O bus 1450 to and from I/O devices 1410, 1430, 1460, 1470 and the processor 1480 and memory 1510, 1520 is controlled by an I/O controller. User input is obtained through a keyboard 1470, mouse, stylus, microphone, trackball, touch-sensitive screen, or other input device. These elements will be found in a conventional desktop or workstation computer as well as other computers suitable for executing computer programs implementing the methods described here, which may be used in conjunction with any digital print engine 1475 or marking engine, display monitor 1460, or other raster output device capable of producing color or gray scale pixels on paper, film, display screen, or other output medium.

[0064] By way of example, a printing device 1475 implementing an interpreter for a page description language, such as the PostScript ® language, includes a microprocessor 1550 for executing program instructions (including font instructions) stored on a printer random access memory (RAM) 1560 and a printer read-only memory (ROM) 1570 and controlling a printer marking engine 1580. The RAM 1560 is optionally supplemented by a mass storage device such as a hard disk (not shown).

[0065] Storage devices suitable for tangibly embodying computer program instructions include all forms of nonvolatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks 1030 and removable disks 1410; magneto-optical disks; and CD-ROM disks. Any of the foregoing may be supplemented by, or incorporated in, specially-designed ASICs (application-specific integrated circuits).

[0066] Although elements of the invention are described in terms of a software implementation, the invention may be implemented in software or hardware or firmware, or a combination of any of the three.

[0067] The present invention has been described in terms of an embodiment. The invention, however, is not limited to the-embodiment depicted and described. Rather, the scope of the invention is defined by the claims.

Claims

1. A method for preparing a resolution-independent vector representation of a region for trapping, comprising:

    finding a peeker between a first edge in the region and a second edge in the region, the peeker being a gap between two edges in vector representation that is narrower than a device pixel at the resolution of the output device when rendered in rasterized form;
    identifying an abutting color where the peeker exists that will abut a side of the first edge between the first edge and the second edge when the first edge is rendered in rasterized form; and
    changing the color of the first edge on the side of the first edge between the first edge and the second edge to the abutting color prior to trapping where the peeker exists.

2. The method of claim 1, wherein:

    the region is a page.

3. The method of claim 1, further comprising:

    rendering a first path and a second path, with the changed color.

4. The method of claim 3, wherein:

    rendering includes trapping.

5. The method of claim 1, wherein finding a peeker comprises:

    deriving from the resolution-independent representation a plurality of objects, each of which is defined by one or more closed directed paths consisting of ordered connected directed straight path edges, each object having an interior sequence number shared by each of the object's edges; and
    finding a peeker between a first edge and a second edge if a distance between the first edge and the second edge is less than a peeker distance.

6. The method of claim 5, wherein:

    the peeker distance is proportional to the height of one pixel of the region represented in rasterized form.

7. The method of claim 6, wherein:

    the peeker distance is approximately twice the height of one pixel of the region represented in rasterized form.

8. The method of claim 5, wherein:

    the peeker distance is proportional to the width of one pixel of the region represented in rasterized form.

9. The method of claim 5, wherein:

    the peeker distance is approximately twice the width of one pixel of the region represented in rasterized form.

10. A method according to claim 1, further comprising a step of identifying an exterior color of a first edge of a first path, comprising:

    if the first edge intersects a second edge of a second path, and if the color of the side of the second edge facing the exterior of the first edge is known, changing the exterior color of the first edge to the color of the side of the second edge facing the exterior of the first edge.

11. The method of claim 10, further comprising:

    changing the color of edges connected to the first edge to the color of the side of the second edge facing the exterior of the first edge.

12. The method of claim 11, further comprising:

    applying another method to identify exterior colors of a third subset of the plurality of visible edges,
    wherein the other method comprises:

        generating a vector from each edge of the first path for which the exterior color is unknown; and
        identifying a second edge of a second path with which the vector intersects on a side of the second edge; and
        changing the exterior color of the first edge to the color of the second edge on the side at which the vector intersects the second edge.

13. The method of claim 12, further comprising:

changing the exterior color of edges connected to the first edge to the color of the second edge on the side at which the vector intersects the second edge.

14. A computer program product tangibly embodied in a machine-readable storage device, said computer program product having program parts for executing the method according to one of claims 1 to 13.

Patentansprüche

1. Verfahren zum Ausbilden einer Auflösungs-unabhängigen Vektordarstellung eines Bereichs zum Einfangen bzw. Trapping , umfassend:

Finden einer Farblücke bzw. eines Peekers zwischen einer ersten Kante bzw. Rand in dem Bereich und einer zweiten Kante bzw. Rand in dem Bereich, wobei die Farblücke bzw. der Peeker einen Spalt zwischen zwei Kanten bzw. Rändern in einer Vektordarstellung ist, welcher schmäler als ein Vorrichtungs-Bildpunkt bzw. Pixsel bei der Auflösung der Ausgabevorrichtung ist, wenn sie in einer gerasterten Form gerendert bzw. wiedergegeben wird;
Identifizieren einer angrenzenden Farbe, wo die Farblücke existiert, welche an einer Seite der ersten Kante zwischen der ersten Kante und der zweiten Kante angrenzen wird, wenn die erste Kante in gerasterter Form gerendert wird; und
Ändern der Farbe der ersten Kante auf der Seite der ersten Kante zwischen der ersten Kante und der zweiten Kante auf die angrenzende Farbe vor einem Einfangen, wo die Farblücke besteht.

2. Verfahren nach Anspruch 1, wobei:

der Bereich eine Seite ist.

3. Verfahren nach Anspruch 1, weiters umfassend:

ein Rendern eines ersten Pfads und eines zweiten Pfads, mit der geänderten Farbe.

4. Verfahren nach Anspruch 3, wobei:

ein Rendern ein Einfangen bzw. Trapping beinhaltet.

5. Verfahren nach Anspruch 1, wobei ein Finden einer Farblücke bzw. eines Peekers umfaßt:

Ableiten aus der Auflösungs-unabhängigen Darstellung einer Mehrzahl von Gegenständen bzw. Objekten, von welchen jeder von diesen durch einen oder mehreren geschlossenen gerichteten Pfad(en) definiert wird, welche(r) aus geordneten, verbundenen, gerichteten, geraden Pfadkanten bzw. -rändern besteht (bestehen), wobei jeder Gegenstand bzw. Objekt eine innere Sequenznummer aufweist, die durch jede der Gegenstands- bzw. Objektkanten geteilt wird; und
Finden einer Farblücke bzw. eines Peekers zwischen einer ersten Kante bzw. Rand und einer zweiten Kante, wenn ein Abstand zwischen der ersten Kante und der zweiten Kante kleiner als ein Peeker- bzw. Farblükkenabstand wird.

6. Verfahren nach Anspruch 5, wobei:

der Farblückenabstand proportional zu der Höhe eines Bildpunkts des Bereichs ist, der in gerasterter Form dargestellt wird.

7. Verfahren nach Anspruch 6, wobei:

der Farblückenabstand etwa zwei Mal der Höhe eines Bildpunkts des Bereichs ist, der in gerasterter Form dargestellt wird.

8. Verfahren nach Anspruch 5, wobei:

der Farblückenabstand proportional der Breite eines Bildpunkts des Bereichs ist, der in gerasterter Form dargestellt wird.

9. Verfahren nach Anspruch 5, wobei:

der Farblückenabstand etwa zwei Mal der Breite eines Bildpunkts des Bereichs ist, der in gerasteter Form dargestellt wird.

10. Verfahren nach Anspruch 1, weiters umfassend einen Schritt eines Identifizierens einer äußeren Farbe einer ersten Kante bzw. Randes eines ersten Pfads, umfassend:

wenn die erste Kante eine zweite Kante bzw. Rand eines zweiten Pfads schneidet, und wenn die Farbe der Seite der zweiten Kante, die zu dem Äußeren der ersten Kante schaut bzw. gerichtet wird bzw. gegenüberliegt, bekannt ist, Verändern der äußeren Farbe der ersten Kante auf die Farbe der Seite der zweiten Kante, die zu dem Äußeren der ersten Kante schaut.

11. Verfahren nach Anspruch 10, weiters umfassend:

Verändern der Farbe von Kanten, die mit der ersten Kante verbunden sind, auf die Farbe der Seite der zweiten Kante, die zu dem Äußeren

der zweiten Kante schaut.

12. Verfahren nach Anspruch 11, weiters umfassend:

Anwenden eines anderen Verfahrens zum Identifizieren von äußeren Farben eines dritten Sub-Satzes aus der Mehrzahl von sichtbaren Kanten bzw. Rändern, wobei das andere Verfahren umfaßt:

Generieren bzw. Erzeugen eines Vektors von jeder Kante des ersten Pfads, für welchen die äußere Farbe unbekannt ist; und Identifizieren einer zweiten Kante bzw. Randes eines zweiten Pfads, mit welchem der Vektor an einer Seite der zweiten Kante schneidet; und
Verändern der äußeren Farbe der ersten Kante auf die Farbe der zweiten Kante auf der Seite, an welcher der Vektor die zweite Kante schneidet.

13. Verfahren nach Anspruch 12, weiters umfassend:

Verändern der äußeren Farbe von Kanten, die mit der ersten Kante verbunden sind, auf die Farbe der zweiten Kante auf der Seite, an welcher der Vektor die zweite Kante schneidet.

14. Computerprogrammprodukt, das greifbar bzw. körperlich in einer durch eine Maschine lesbaren Speichervorrichtung verkörpert ist, wobei das Computerprogramm Produktprogrammteile zum Ausführen des Verfahrens nach einem der Ansprüche 1 bis 13 aufweist.

Revendications

1. Procédé de préparation d'une représentation vectorielle indépendante de la résolution d'une région à piéger, comprenant les étapes suivantes :

on repère un interstice ou peeker entre un premier bord de la région et un second bord de la région, l'interstice étant un intervalle entre deux bords de la représentation vectorielle qui est plus étroit qu'un pixel de périphérique à la résolution du périphérique de sortie lorsqu'il est restitué sous forme rastérisée;
on identifie une couleur attenante à l'endroit où l'interstice existe, qui s'appuiera sur un côté du premier bord entre le premier bord et le second bord lorsque le premier bord est restitué sous forme rastérisée; et
on fait passer la couleur du premier bord sur le côté du premier bord entre le premier bord et le second bord à la couleur attenante avant de piéger l'endroit où l'interstice existe.

2. Procédé selon la revendication 1, dans lequel la région est une page.

3. Procédé selon la revendication 1, comprenant en outre :

l'étape de restitution d'un premier trajet et d'un second trajet, avec la couleur changée.

4. Procédé selon la revendication 3, dans lequel la restitution inclut le piégeage.

5. Procédé selon la revendication 1, dans lequel le repérage d'un interstice comprend :

la dérivation d'une pluralité d'objets à partir de la représentation indépendante de la résolution, chacun d'eux étant défini par un ou plusieurs trajets dirigés fermés constitués de bords de trajets droits dirigés connectés ordonnés, chaque objet ayant un nombre de séquences internes partagées par chacun des bords de l'objet; et
le repérage d'un interstice entre un premier bord et un second bord si une distance entre le premier bord et le second bord est inférieure à une distance d'interstice.

6. Procédé selon la revendication 5, dans lequel :

la distance d'interstice est proportionnelle à la hauteur d'un pixel de la région représentée sous forme rastérisée.

7. Procédé selon la revendication 6, dans lequel :

la distance d'interstice est approximativement de deux fois la hauteur d'un pixel de la région représentée sous forme rastérisée.

8. Procédé selon la revendication 5, dans lequel :

la distance d'interstice est proportionnelle à la largeur d'un pixel de la région représentée sous forme rastérisée.

9. Procédé selon la revendication 5, dans lequel :

la distance d'interstice est approximativement de deux fois la largeur d'un pixel de la région représentée sous forme rastérisée.

10. Procédé selon la revendication 1, comprenant en outre une étape d'identification d'une couleur extérieure d'un premier bord d'un premier trajet, comprenant :

si le premier bord coupe un second bord d'un second trajet et si la couleur du côté du second bord faisant face à l'extérieur du premier bord est connue, le passage de la couleur extérieure du premier bord à la couleur du côté du second bord faisant face à l'extérieur du premier bord.

11. Procédé selon la revendication 10, comprenant en outre :

le passage de la couleur de bords connectés au premier bord à la couleur du côté du second bord faisant face à l'extérieur du premier bord.

12. Procédé selon la revendication 11, comprenant en outre :

l'application d'un autre procédé pour identifier les couleurs extérieures d'un troisième sous-ensemble de la pluralité de bords visibles,

dans lequel l'autre procédé comprend :

la génération d'un vecteur à partir de chaque bord du premier trajet pour lequel la couleur extérieure est inconnue; et
l'identification d'un second bord d'un second trajet que le vecteur coupe sur un côté du second bord; et
le passage de la couleur extérieure du premier bord à la couleur du second bord sur le côté sur lequel le vecteur coupe le second bord.

13. Procédé selon la revendication 12, comprenant en outre :

la passage de la couleur extérieure de bords connectés au premier bord à la couleur du second bord sur le côté sur lequel le vecteur coupe le second bord.

14. Produit de programme informatique mis en oeuvre de manière tangible dans un dispositif de stockage lisible en machine, ledit produit de programme informatique ayant des parties de programme pour exécuter le procédé selon l'une quelconque des revendications 1 à 13.

FIG. 1

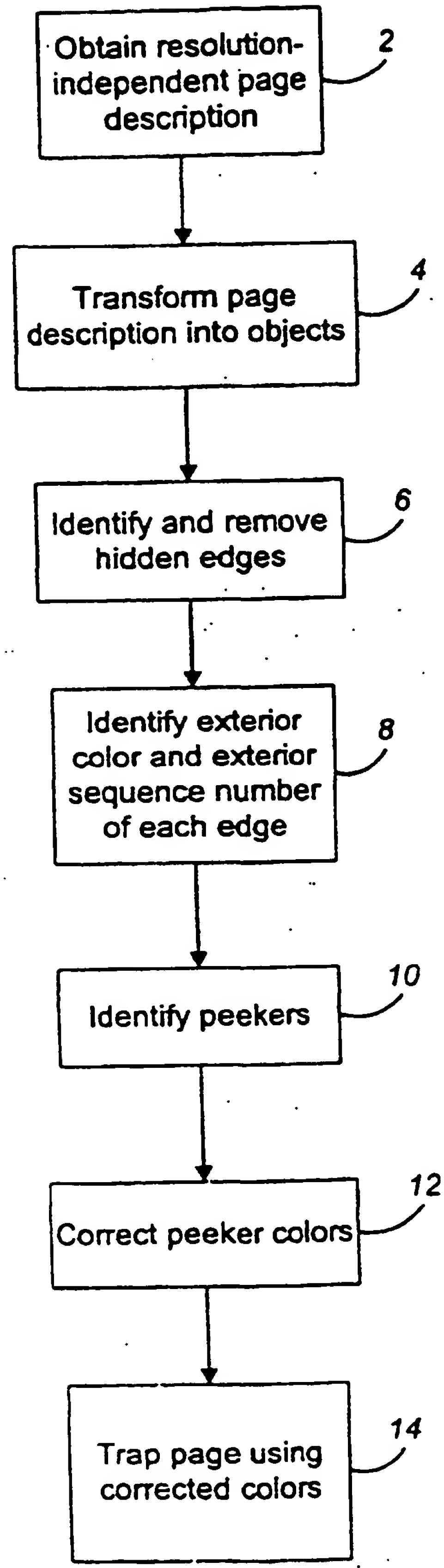

Obtain resolution-independent page description
2
Transform page description into objects
4
Identify and remove hidden edges
6
Identify exterior color and exterior sequence number of each edge
8
Identify peekers
10
Correct peeker colors
12
Trap page using corrected colors
14

FIG. 2a

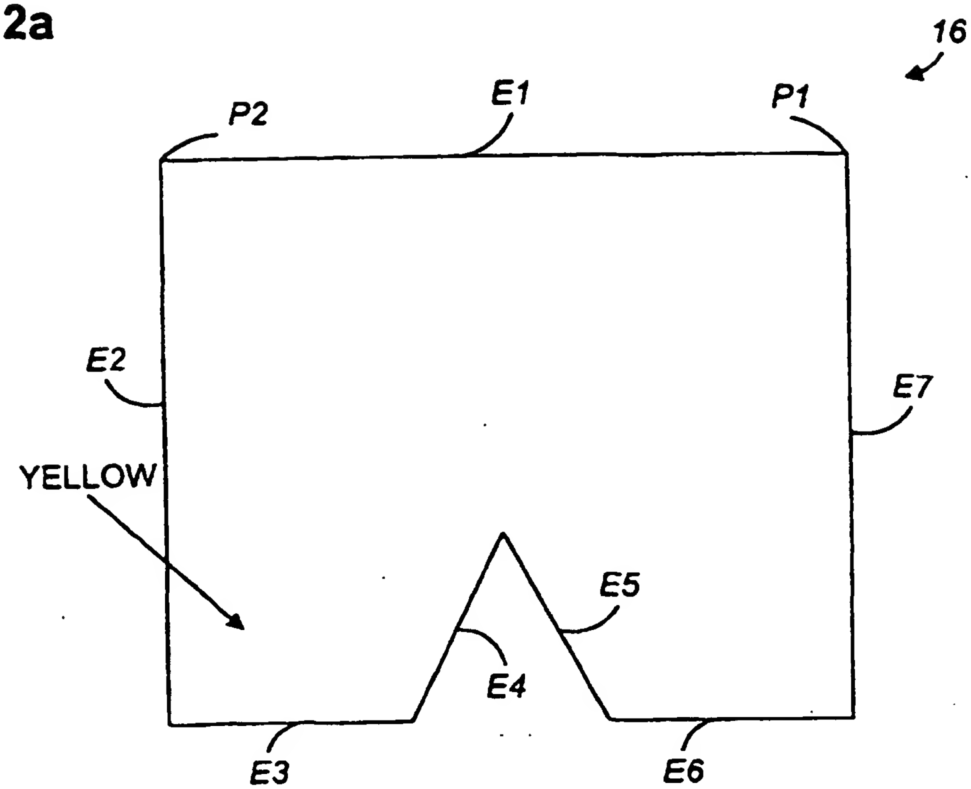

16
P2
E1
P1
E2
E7
YELLOW
E5
E4
E3
E6

FIG. 2b

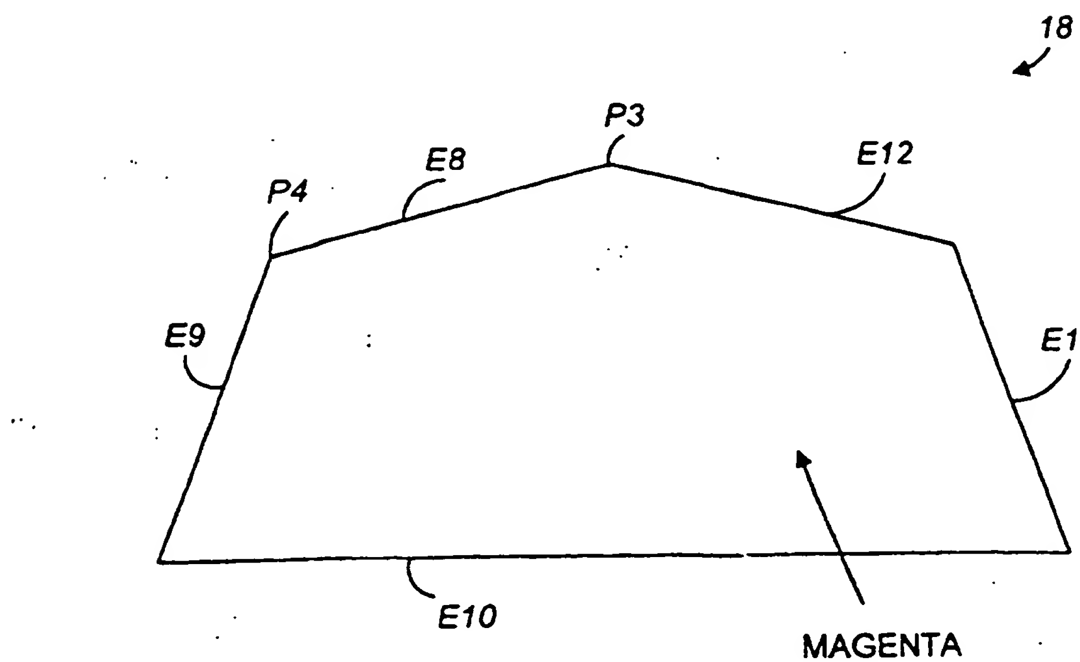

18
P3
E8
E12
P4
E9
E11
E10
MAGENTA

FIG. 2c

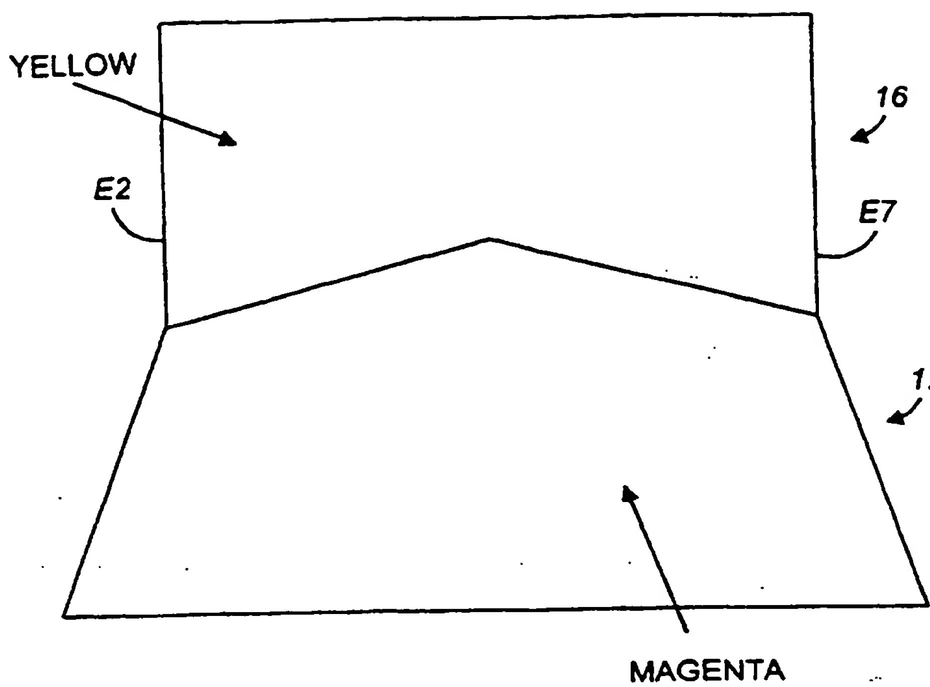

YELLOW
16
E2
E7
12
MAGENTA

FIG. 2d

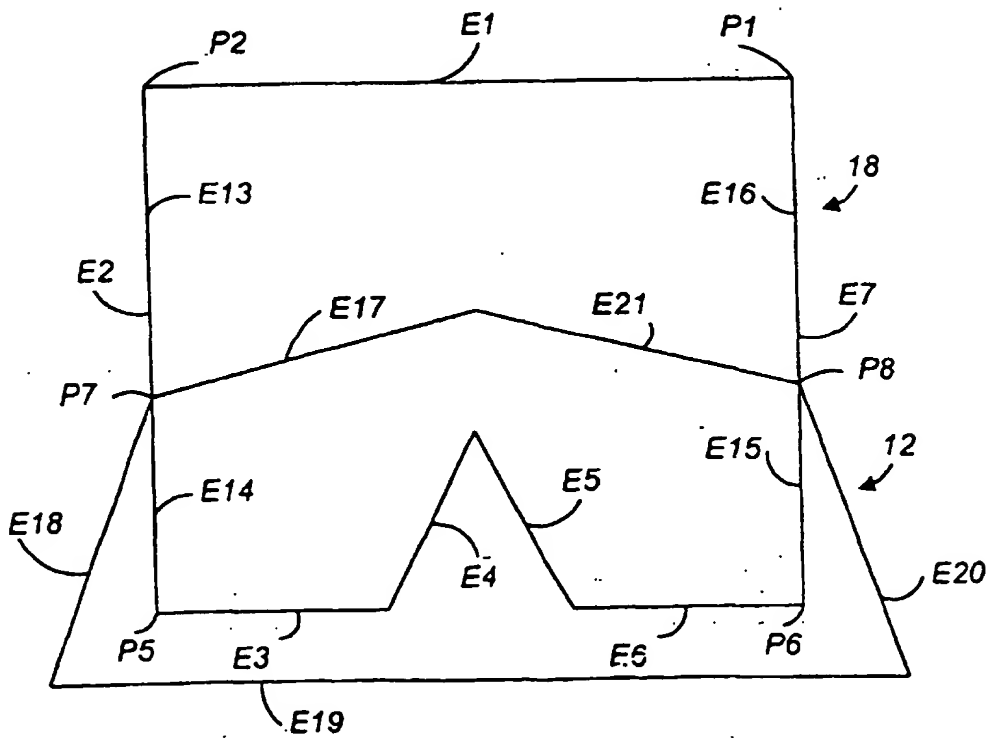

P2
E1
P1
18
E13
E16
E2
E17
E21
E7
P7
P8
E15
12
E14
E5
E18
E4
E20
P5
E3
E6
P6
E19

FIG. 2e

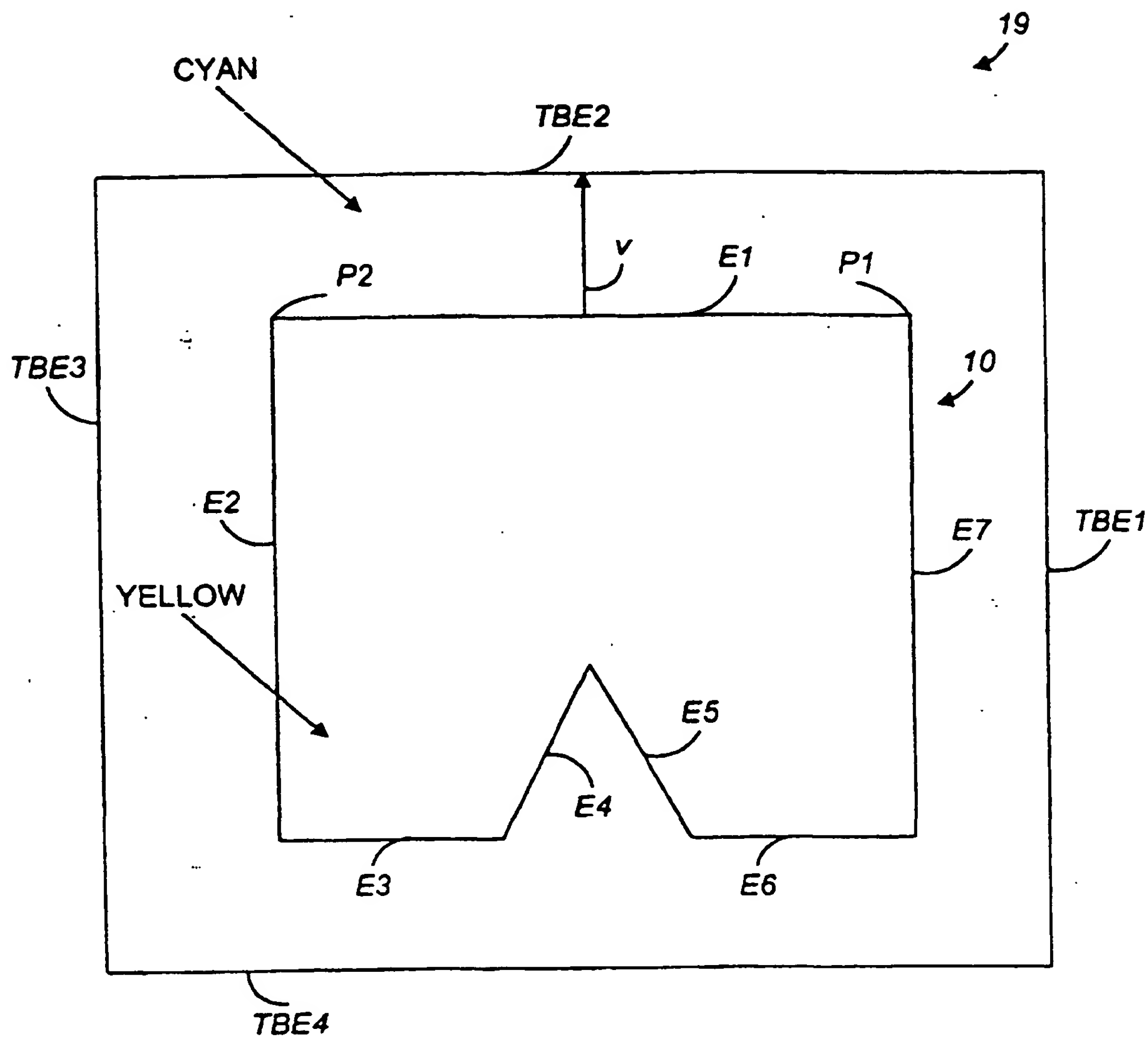


FIG. 3a

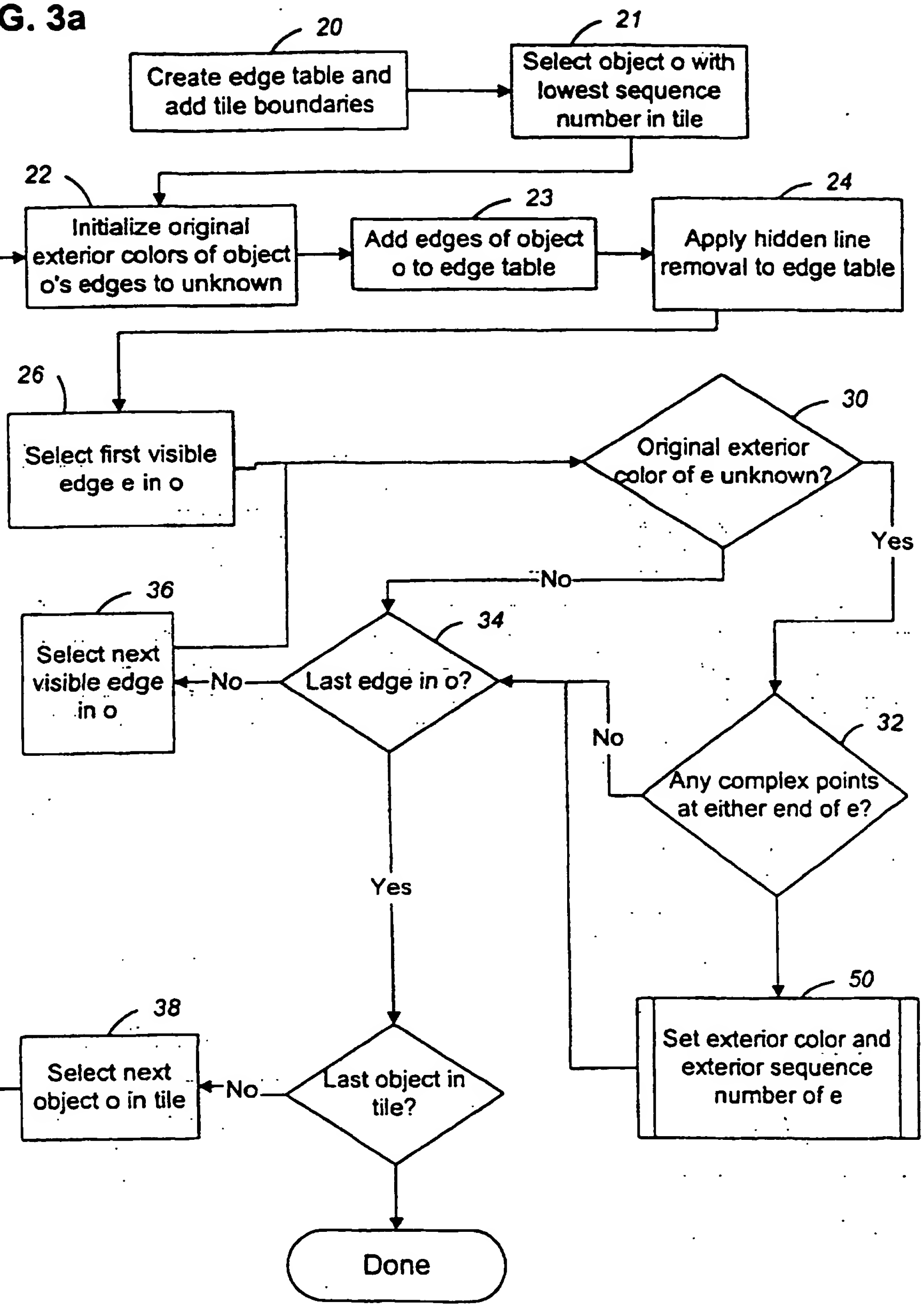

20
Create edge table and add tile boundaries
21
Select object o with lowest sequence number in tile
22
Initialize original exterior colors of object o's edges to unknown
23
Add edges of object o to edge table
24
Apply hidden line removal to edge table
26
Select first visible edge e in o
30
Original exterior color of e unknown?
Yes
No
36
Select next visible edge in o
No
34
Last edge in o?
No
32
Any complex points at either end of e?
Yes
50
Set exterior color and exterior sequence number of e
38
Select next object o in tile
No
Last object in tile?
Done

FIG. 3b

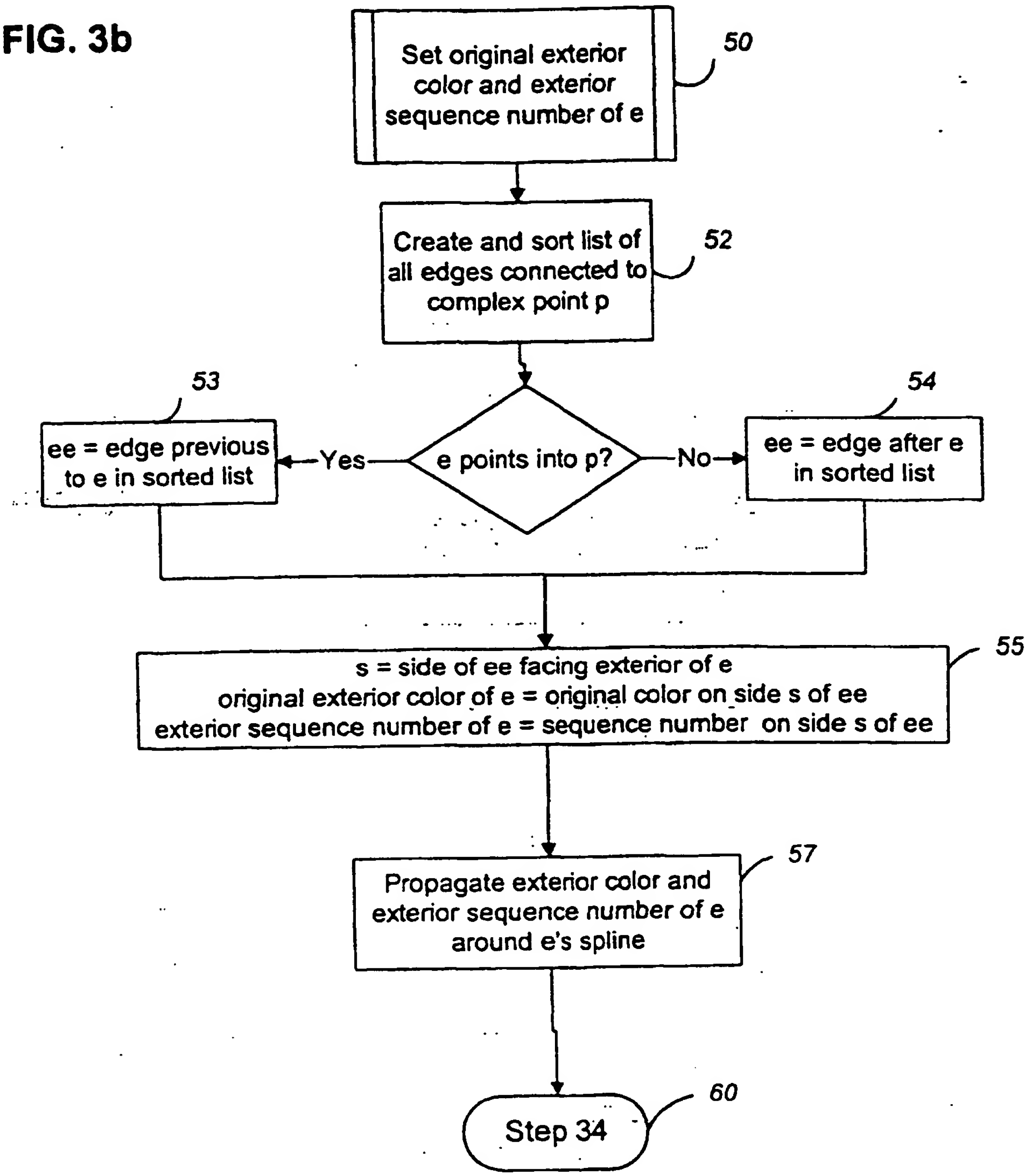

Set original exterior color and exterior sequence number of e
50
Create and sort list of all edges connected to complex point p
52
e points into p?
Yes
No
ee = edge previous to e in sorted list
53
ee = edge after e in sorted list
54
s = side of ee facing exterior of e
original exterior color of e = original color on side s of ee
exterior sequence number of e = sequence number on side s of ee
55
Propagate exterior color and exterior sequence number of e around e's spline
57
Step 34
60

FIG. 3c

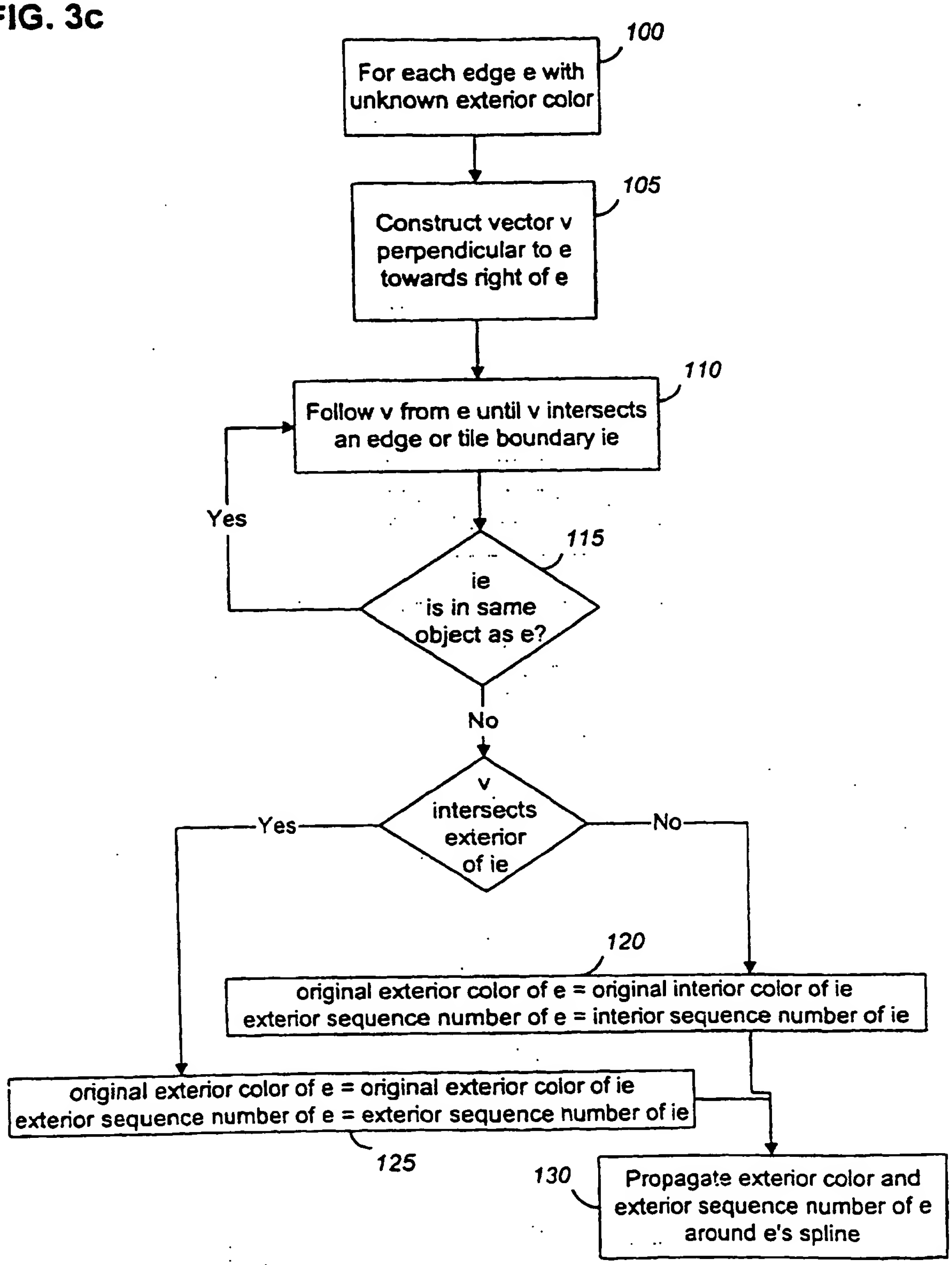

100
For each edge e with unknown exterior color
105
Construct vector v perpendicular to e towards right of e
110
Follow v from e until v intersects an edge or tile boundary ie
Yes
115
ie is in same object as e?
No
v intersects exterior of ie
Yes
No
120
original exterior color of e = original interior color of ie
exterior sequence number of e = interior sequence number of ie
original exterior color of e = original exterior color of ie
exterior sequence number of e = exterior sequence number of ie
125
130
Propagate exterior color and exterior sequence number of e around e's spline

FIG. 4a

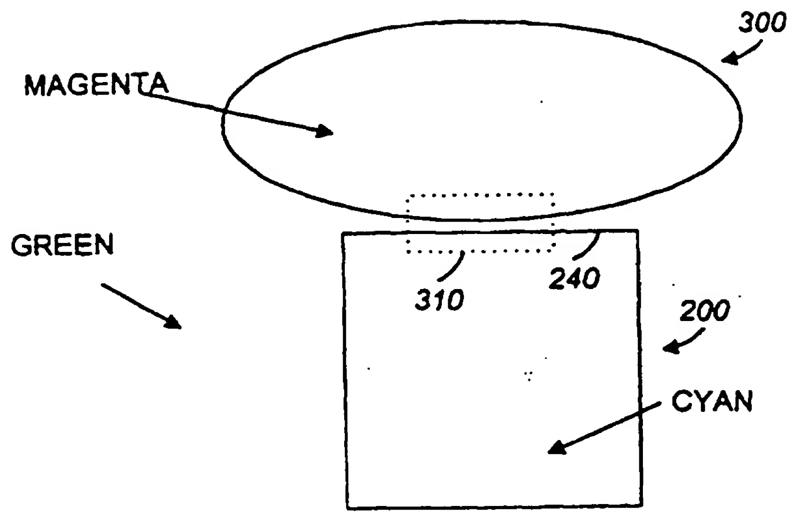


FIG. 4b

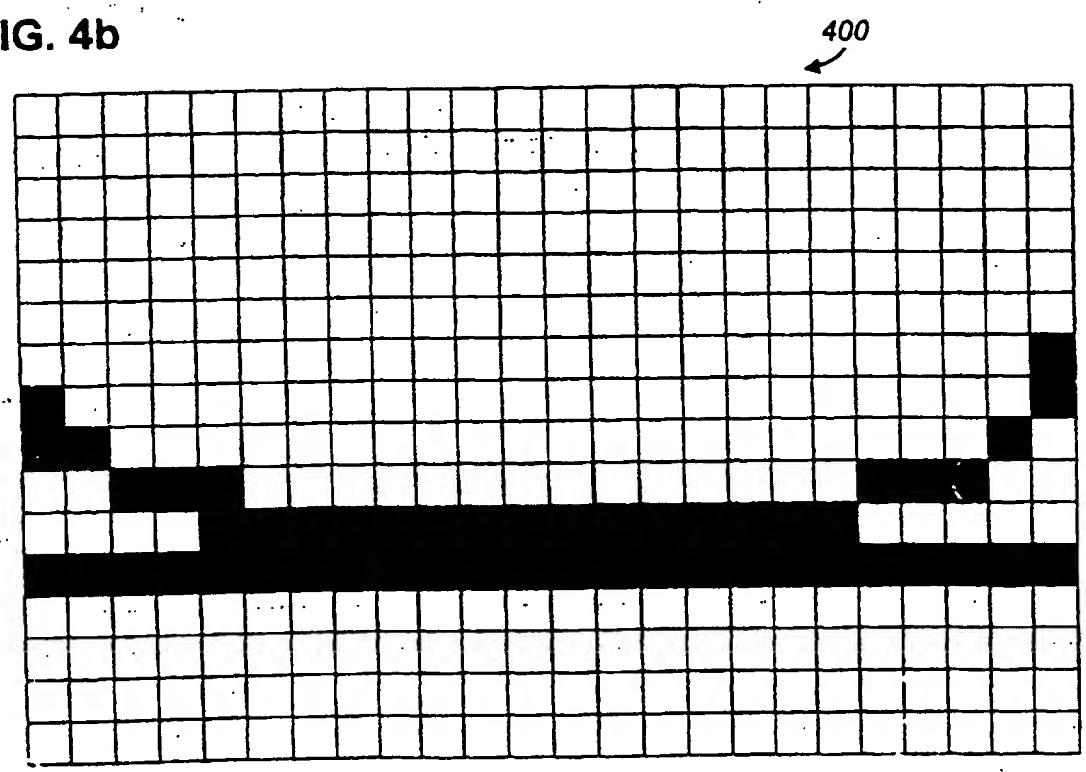
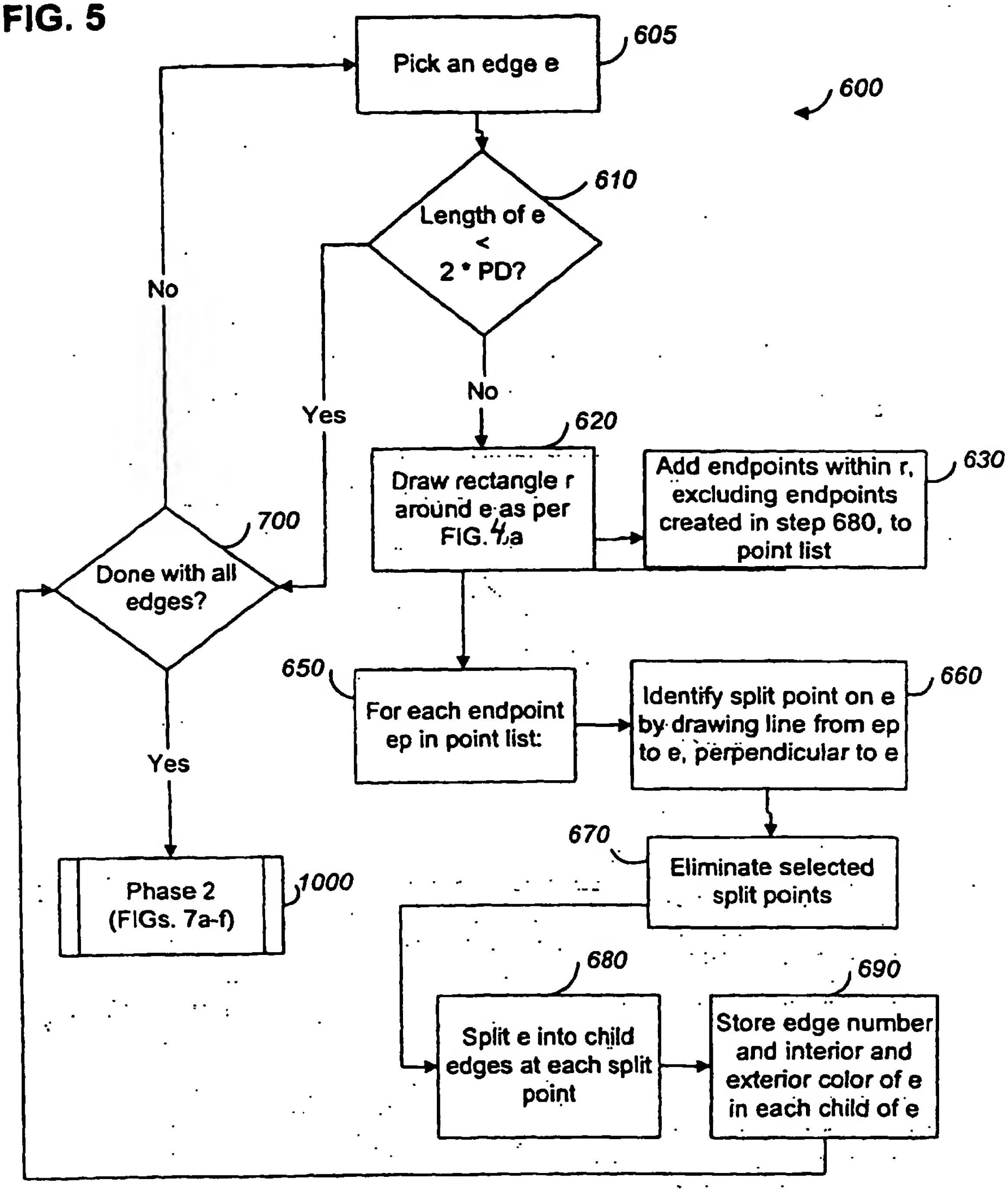


FIG. 5



605
Pick an edge e
600
610
Length of e < 2 * PD?
No
Yes
No
620
Draw rectangle r around e as per FIG. 4a
630
Add endpoints within r, excluding endpoints created in step 680, to point list
700
Done with all edges?
650
For each endpoint ep in point list:
660
Identify split point on e by drawing line from ep to e, perpendicular to e
Yes
670
Eliminate selected split points
Phase 2 (FIGs. 7a-f)
1000
680
690
Split e into child edges at each split point
Store edge number and interior and exterior color of e in each child of e

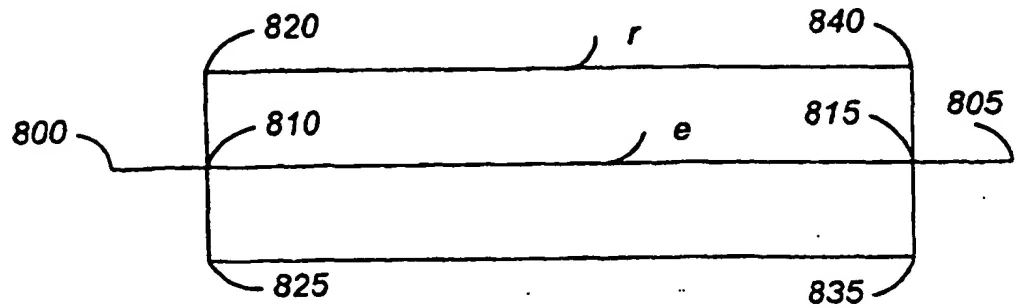


FIG. 6a

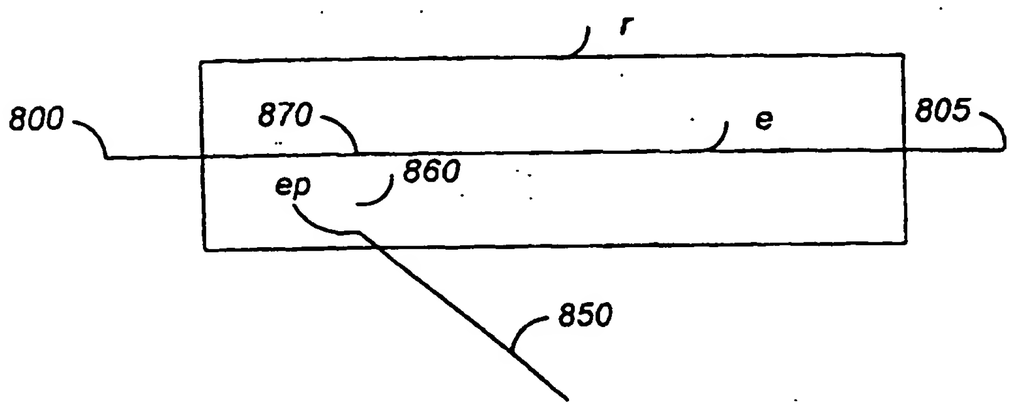


FIG. 6b

FIG. 7a

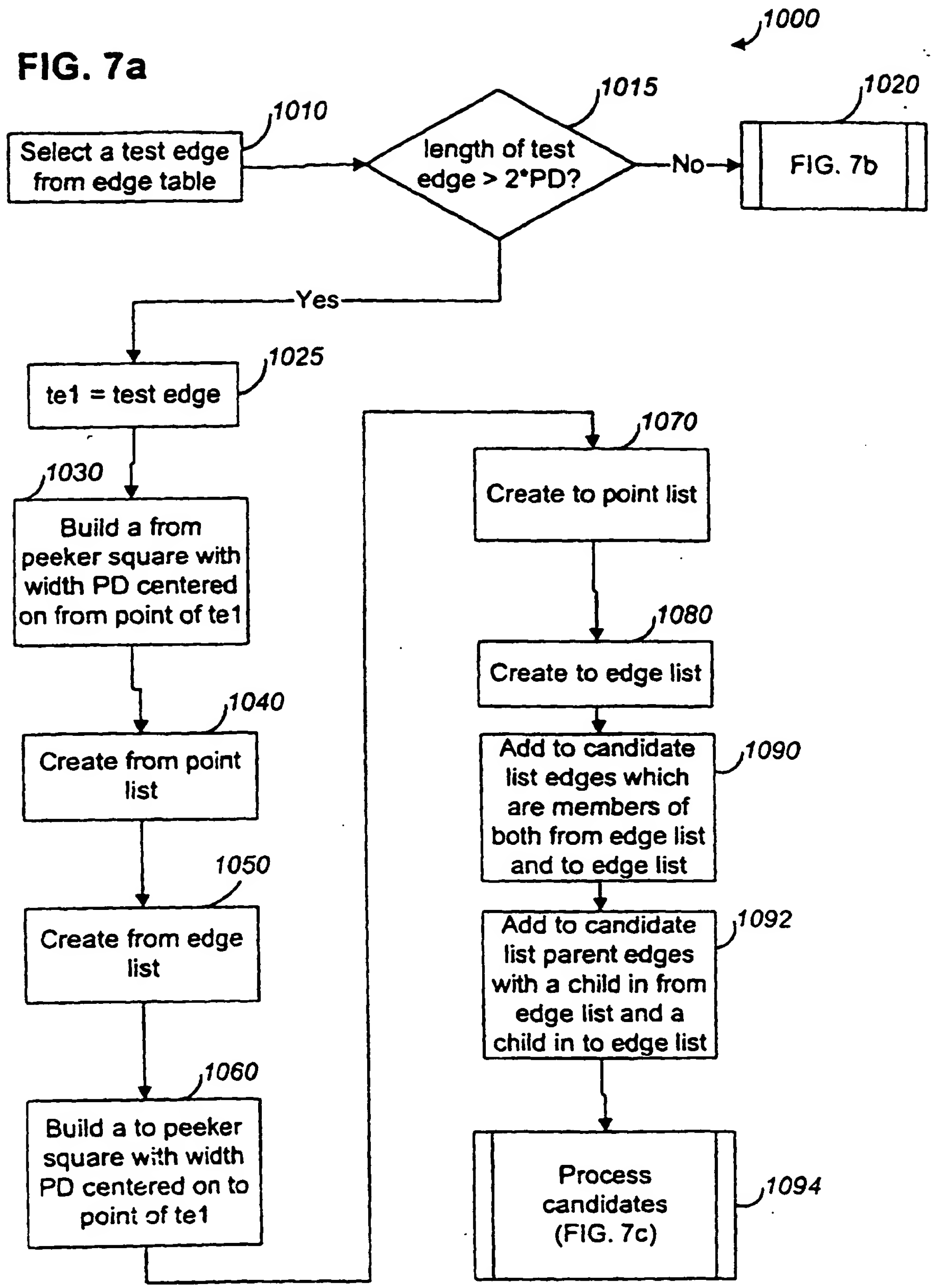

1000
1010
Select a test edge from edge table
1015
length of test edge > 2*PD?
No
1020
FIG. 7b
Yes
1025
te1 = test edge
1030
Build a from peeker square with width PD centered on from point of te1
1040
Create from point list
1050
Create from edge list
1060
Build a to peeker square with width PD centered on to point of te1
1070
Create to point list
1080
Create to edge list
1090
Add to candidate list edges which are members of both from edge list and to edge list
1092
Add to candidate list parent edges with a child in from edge list and a child in to edge list
1094
Process candidates (FIG. 7c)

FIG. 7b

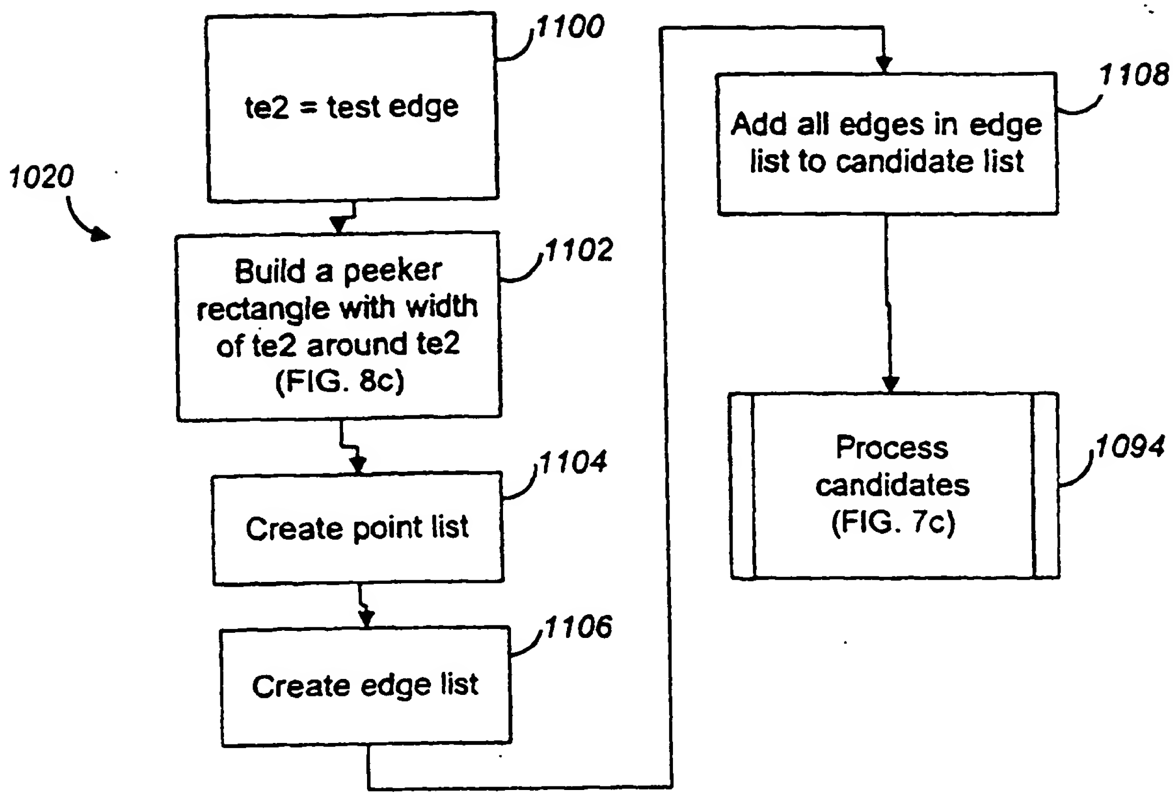

1100
te2 = test edge
1020
Build a peeker rectangle with width of te2 around te2 (FIG. 8c)
1102
Create point list
1104
Create edge list
1106
Add all edges in edge list to candidate list
1108
Process candidates (FIG. 7c)
1094

FIG. 7c

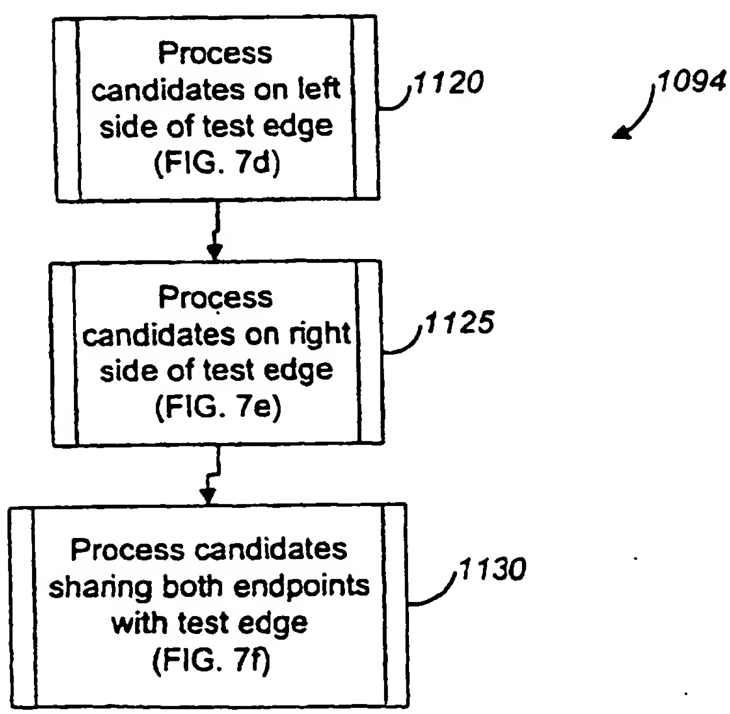

Process candidates on left side of test edge (FIG. 7d)
1120
1094
Process candidates on right side of test edge (FIG. 7e)
1125
Process candidates sharing both endpoints with test edge (FIG. 7f)
1130

FIG. 7d

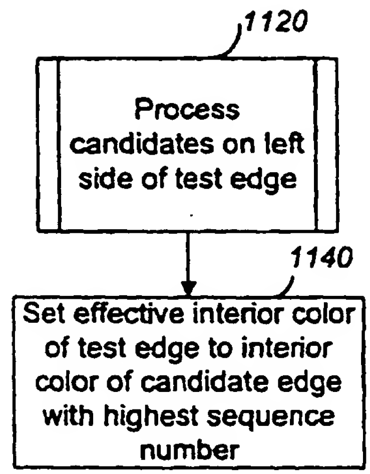

1120
Process candidates on left side of test edge
1140
Set effective interior color of test edge to interior color of candidate edge with highest sequence number

FIG. 7e

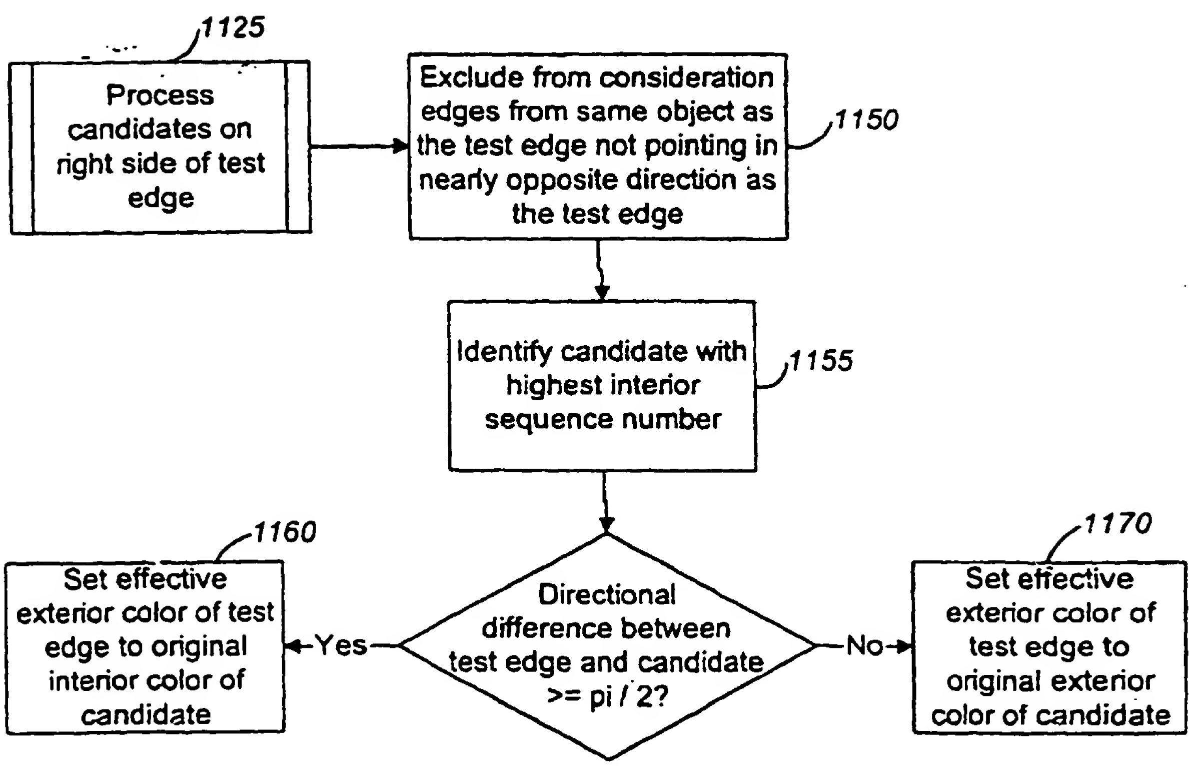
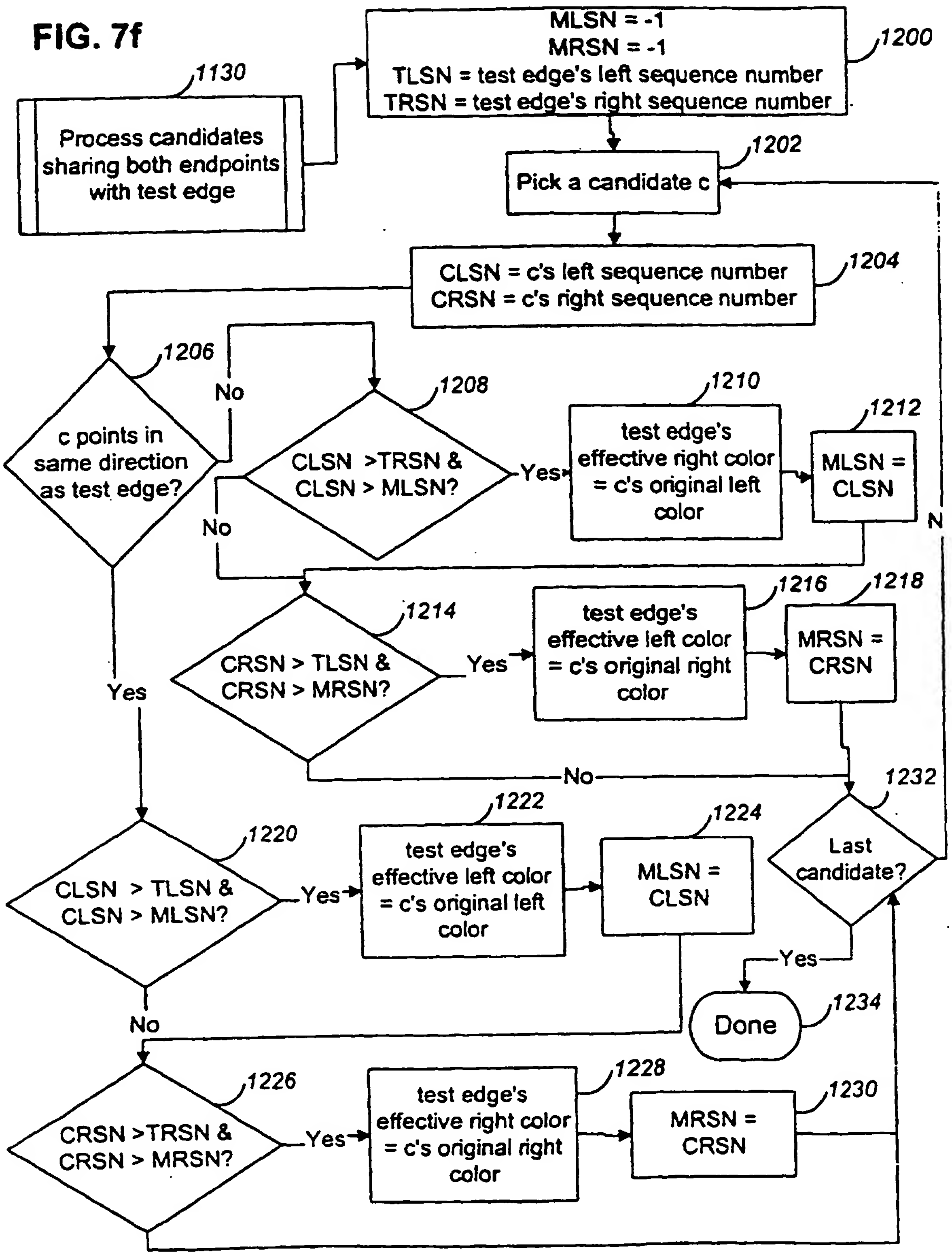

1125
Process candidates on right side of test edge
Exclude from consideration edges from same object as the test edge not pointing in nearly opposite direction as the test edge
1150
Identify candidate with highest interior sequence number
1155
1160
Set effective exterior color of test edge to original interior color of candidate
Yes
Directional difference between test edge and candidate >= pi / 2?
No
1170
Set effective exterior color of test edge to original exterior color of candidate

FIG. 7f



1130
Process candidates sharing both endpoints with test edge
1200
MLSN = -1
MRSN = -1
TLSN = test edge's left sequence number
TRSN = test edge's right sequence number
1202
Pick a candidate c
1204
CLSN = c's left sequence number
CRSN = c's right sequence number
1206
c points in same direction as test edge?
No
Yes
1208
CLSN >TRSN & CLSN > MLSN?
Yes
No
1210
test edge's effective right color = c's original left color
1212
MLSN = CLSN
N
1214
CRSN > TLSN & CRSN > MRSN?
Yes
No
1216
test edge's effective left color = c's original right color
1218
MRSN = CRSN
1232
Last candidate?
Yes
1234
Done
1220
CLSN > TLSN & CLSN > MLSN?
Yes
No
1222
test edge's effective left color = c's original left color
1224
MLSN = CLSN
1226
CRSN >TRSN & CRSN > MRSN?
Yes
1228
test edge's effective right color = c's original right color
1230
MRSN = CRSN

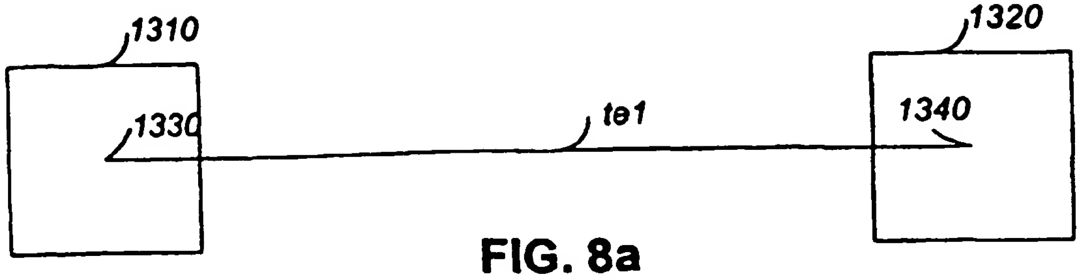


FIG. 8a

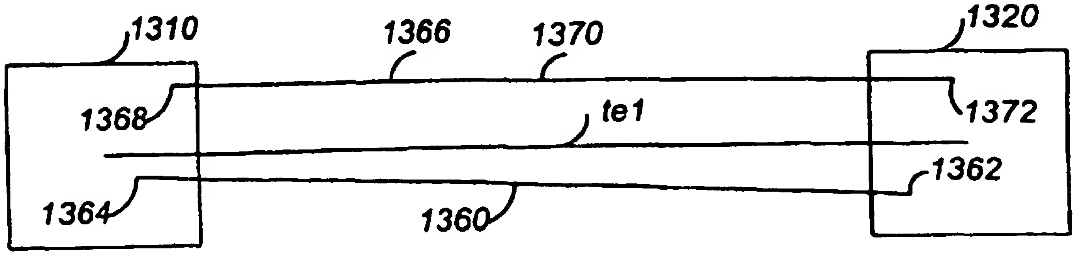


FIG. 8b

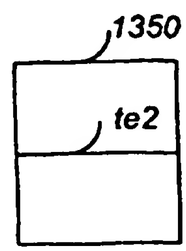


FIG. 8c

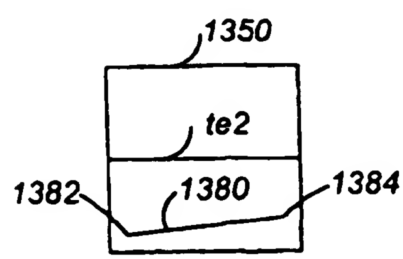
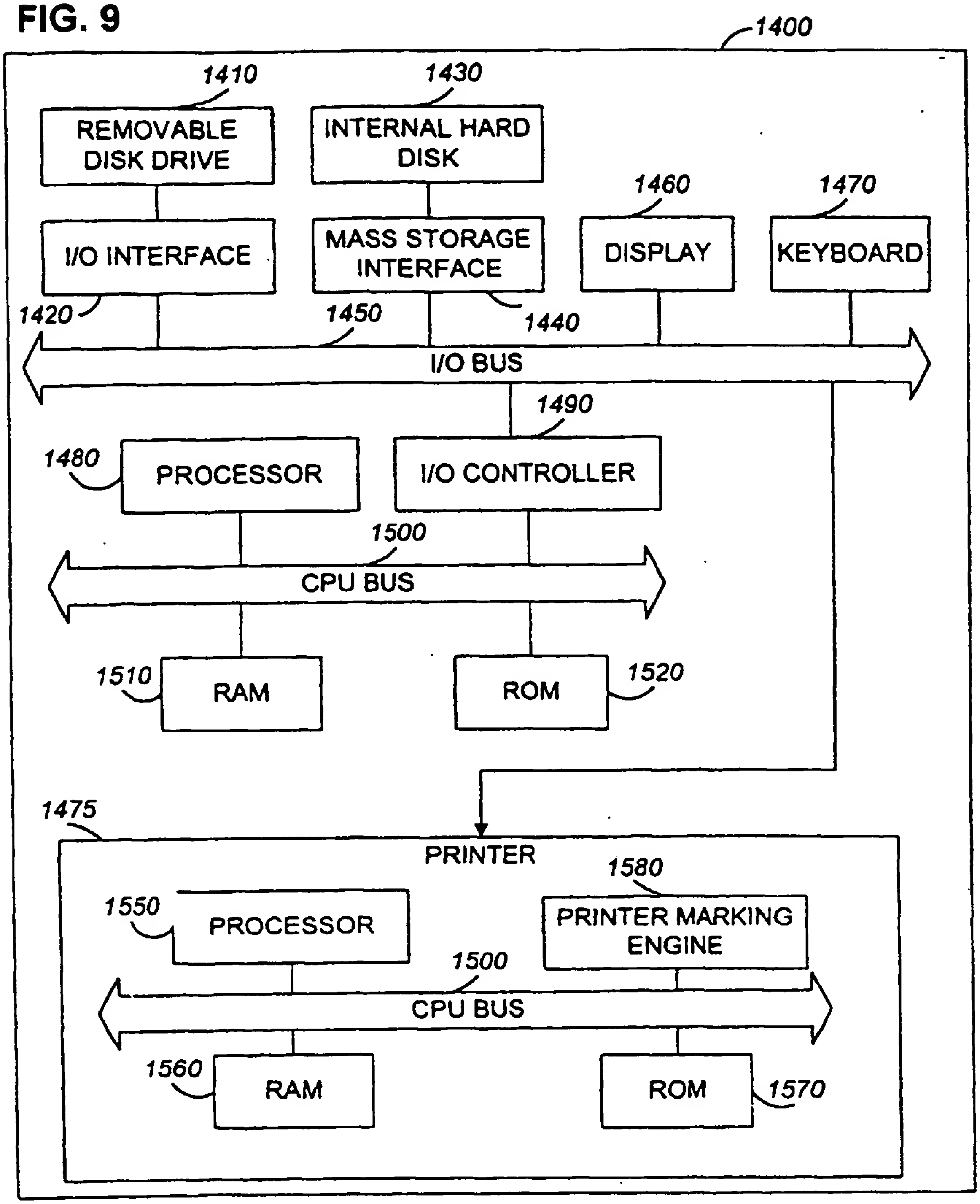


FIG. 8d

FIG. 9


1400
1410
REMOVABLE DISK DRIVE
1430
INTERNAL HARD DISK
1460
1470
I/O INTERFACE
MASS STORAGE INTERFACE
DISPLAY
KEYBOARD
1420
1450
1440
I/O BUS
1490
1480
PROCESSOR
I/O CONTROLLER
1500
CPU BUS
1510
RAM
ROM
1520
1475
PRINTER
1580
1550
PROCESSOR
PRINTER MARKING ENGINE
1500
CPU BUS
1560
RAM
ROM
1570